Projective Integration for Moment Equations

Julian Koellermeier

Peking University

November 21st, 2018

Today's Topic

- Quadrature-Based Moment Equations
- Filtered Hyperbolic Moment Equations
- Projective Integration for Moment Equations
- Hyperbolic Shallow Water Moment Equations
- Adaptive Moment Model

Today's Topic

- Quadrature-Based Moment Equations
- Filtered Hyperbolic Moment Equations
- **Projective Integration for Moment Equations**
- Hyperbolic Shallow Water Moment Equations
- Adaptive Moment Model

Stiff PDEs/ODEs

Consider the following PDE

$$\partial_t u + \partial_x F(u) = -\frac{1}{\epsilon} S(u), \quad x \in \Omega, t \in \mathbb{R}^+$$

$$\Rightarrow \quad \partial_t u = -\frac{1}{\epsilon} S(u) - \partial_x F(u)$$

Spatial discretization $x_1, x_2, \ldots, x_N$, for $N \in \mathbb{N}$ using $\boldsymbol{u} = (u_1, u_2, \ldots, u_N)$

$$\partial_t \boldsymbol{u} = D_t(\boldsymbol{u})$$

Stiff PDEs/ODEs

Consider the following PDE

$$\partial_t u + \partial_x F(u) = -\frac{1}{\epsilon} S(u), \quad x \in \Omega, t \in \mathbb{R}^+$$

$$\Rightarrow \quad \partial_t u = -\frac{1}{\epsilon} S(u) - \partial_x F(u)$$

Spatial discretization $x_1, x_2, \dots, x_N$, for $N \in \mathbb{N}$ using $\boldsymbol{u} = (u_1, u_2, \dots, u_N)$

$$\partial_t \boldsymbol{u} = D_t(\boldsymbol{u})$$

In this talk I want to discuss a numerical method for stiff problems

Definition of stiff problem

Consider the following system for system matrix $\boldsymbol{A} \in \mathbb{R}^{M \times M}$ and force $\boldsymbol{f} \in \mathbb{R}^{M}$

$$\partial_t \boldsymbol{u} = \boldsymbol{A}\boldsymbol{u} + \boldsymbol{f}\,(t)$$

The solution is given by

$$\boldsymbol{y}\,(t) = \sum_{i=1}^{M} c_i \exp\left(\lambda_i t\right) \boldsymbol{v}_i + \boldsymbol{g}(t)$$

for $c_i = const$, eigenvalues λ_i, eigenvectors $\boldsymbol{v}_i$ and some stationary solution $\boldsymbol{g}(t)$.
The solution will approach the steady state solution as $t \to \infty$.

Definition of stiff problem

Consider the following system for system matrix $\boldsymbol{A} \in \mathbb{R}^{M\times M}$ and force $\boldsymbol{f} \in \mathbb{R}^{M}$

$$\partial_t \boldsymbol{u} = \boldsymbol{A}\boldsymbol{u} + \boldsymbol{f}\,(t)$$

The solution is given by

$$\boldsymbol{y}\,(t) = \sum_{i=1}^{M} c_i \exp\left(\lambda_i t\right) \boldsymbol{v}_i + \boldsymbol{g}(t)$$

for $c_i = const$, eigenvalues λ_i, eigenvectors $\boldsymbol{v}_i$ and some stationary solution $\boldsymbol{g}(t)$.
The solution will approach the steady state solution as $t \to \infty$.

Decay of each component might be different, depending on $Re\left(\lambda_i\right)$.

Definition of stiff problem

Define the stiffness ratio

$$\frac{max|Re\left(\lambda_i\right)|}{min|Re\left(\lambda_i\right)|}$$

Large stiffness ratio can lead to stability problems.

Stiffness property

- large stiffness ratio
- some components of the solution decay much faster than others
- stability limits time step size Δt instead of accuracy

Examples for stiff ODEs

- Chemical reactions with different reaction speeds

$$
\begin{aligned}
A + X &\xrightarrow{k_1} 2X \\
X + Y &\xrightarrow{k_2} 2 \\
Y &\xrightarrow{k_3} B
\end{aligned}
$$

- Combustion problems
- Fluid dynamics including sonic waves versus bulk movement of flow
- Kinetic equation

$$
\partial_t \boldsymbol{u} + \boldsymbol{A}_M \partial_x \boldsymbol{u} = -\frac{1}{\epsilon} \boldsymbol{S}(\boldsymbol{u})
$$

Explicit vs Implicit schemes

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t \cdot D_t\left(\boldsymbol{u}^n\right)$$

Explicit schemes, e.g. Forward Euler (FE)

+ explicit update formula
+ straightforward implementation
- restrictive time step constraint (no A-stability)

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t \cdot D_t\left(\boldsymbol{u}^{n+1}\right)$$

Implicit schemes, e.g. Implicit Euler (IE)

- require solution of (non-)linear system $\Rightarrow$ slow
- more difficult to implement
+ no time step constraint (A-stability)

Example: Runge-Kutta Scheme

$$\partial_t u = D_t(u)$$

Compute solution using

$$u^{n+1} = u^n + \Delta t \sum_{j=1}^{s} b_j k_j$$

Using right-hand side evaluations

$$k_j = D_t\left(u^n + \Delta t \sum_{l=1}^{s} a_{jl} k_l\right), \quad j = 1, \dots, s.$$

with s the number of steps, Δt the time step size, and a_{jl}, b_j coefficients for evaluations/update at times c_j.

Butcher tableau

$$
\begin{array}{c|c} \mathbf{c} & A \\ \hline & \mathbf{b}^T \end{array} = \begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array}
$$

c_i evaluating times

a_{jl} coefficients for computation of evaluations k_j

b_j coefficients for computation of next time step u^{n+1}

If $\boldsymbol{A}$ is lower diagonal matrix $\Rightarrow$ explicit scheme

Diagonally Implicit Runge Kutta (DIRK) method

$$
\begin{array}{c|ccccc}
c_1 & a_{11} & 0 & \dots & \dots & 0 \\
c_2 & a_{21} & a_{22} & 0 & \dots & 0 \\
\vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\
\vdots & \vdots & \vdots & & \ddots & 0 \\
c_s & a_{s1} & a_{s2} & \dots & \dots & a_{ss} \\
\hline
 & b_1 & b_2 & \dots & \dots & b_s
\end{array}
$$

Upper diagonal matrix is zero.
Only diagonal entries are implicit.

Implicit-explicit RK methods, [PARESCHI, RUSSO, 2005]

Idea: Split ODE into stiff and non-stiff parts

$$\partial_t u = -\frac{1}{\epsilon} S(u) - \partial_x F(u)$$

$$\partial_t u = D_t^{IM}(u) + D_t^{EX}(u)$$

$$u^{n+1} = u^n + \Delta t \sum_{j=1}^{s} \widetilde{b}_j D_t^{EX}(u_j) + \Delta t \sum_{j=1}^{s} b_j D_t^{IM}(u_j)$$

Using intermediate values

$$u_l = u^n + \Delta t \sum_{j=1}^{l-1} \widetilde{a}_{lj} D_t^{EX}(u_j) + \Delta t \sum_{j=1}^{s} a_{lj} D_t^{IM}(u_j)$$

Implicit-explicit (IMEX) methods

$$u^{n+1} = u^n + \Delta t \sum_{j=1}^{s} \widetilde{b}_j D_t^{EX}(u_j) + \Delta t \sum_{j=1}^{s} b_j D_t^{IM}(u_j)$$

$$u_l = u^n + \Delta t \sum_{j=1}^{l-1} \widetilde{a}_{lj} D_t^{EX}(u_j) + \Delta t \sum_{j=1}^{s} a_{lj} D_t^{IM}(u_j)$$

Use two Butcher tableaus

$$\begin{array}{c|c} \widetilde{\boldsymbol{c}} & \widetilde{\boldsymbol{A}} \\ \hline & \widetilde{\boldsymbol{b}}^T \end{array} \text{ and } \begin{array}{c|c} \boldsymbol{c} & \boldsymbol{A} \\ \hline & \boldsymbol{b}^T \end{array}$$

Implicit scheme is usually a DIRK scheme.

Asymptotic Preserving (AP) property [JIN, 1999]

$$\partial_t \boldsymbol{u} + \boldsymbol{A}_M \partial_x \boldsymbol{u} = -\frac{1}{\epsilon} \boldsymbol{S}(\boldsymbol{u}) \tag{1}$$

System depends on parameter ϵ, but for $\epsilon \to 0$ the system converges to a limit equation (e.g. Euler equation).

Definition: Asymptotic-Preserving (AP)

A scheme for problem (1) with discretization parameter Δt is called **Asymptotic Preserving** (AP) if its stability requirement on Δt is independent of ϵ and its limit $\epsilon \to 0$ is consistent with the limit solution.

Usually, small ϵ makes equation stiffer
$\Rightarrow$ smaller Δt is needed for explicit schemes $\Rightarrow$ not AP.

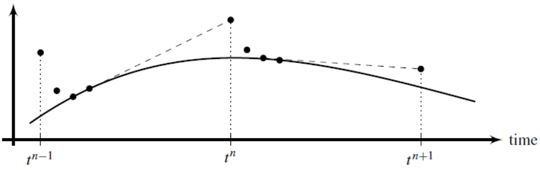
Projective Integration for Moment Equations

joint work with
Giovanni Samaey, KU Leuven

Projective Integration (PI), [KEVREKIDIS, 2003]

Idea:

1. Perform $K+1$ small time steps δt first to damp fast modes.
2. Then extrapolate derivative with one large step Δt.



0. initialize: $\boldsymbol{u}^{n,0} = \boldsymbol{u}^n$
1. for $k = 0, 1, \ldots, K$: $\quad \boldsymbol{u}^{n,k+1} = \boldsymbol{u}^{n,k} + \delta t \, D_t \left(\boldsymbol{u}^{n,k} \right)$
2. extrapolate: $\quad \boldsymbol{u}^{n+1} = \boldsymbol{u}^{n,K+1} + (\Delta t - (K+1)\delta t) \cdot \frac{\boldsymbol{u}^{n,K+1} - \boldsymbol{u}^{n,K}}{\delta t}$

Projective Integration as Runge-Kutta method

Standard definition:

$$
\begin{aligned}
\boldsymbol{u}^{n,1} &= \boldsymbol{u}^{n,0} + \delta t D_t \left(\boldsymbol{u}^{n,0}\right) \\
\boldsymbol{u}^{n,2} &= \boldsymbol{u}^{n,1} + \delta t D_t \left(\boldsymbol{u}^{n,1}\right) \\
&\vdots \\
\boldsymbol{u}^{n,K+1} &= \boldsymbol{u}^{n,K} + \delta t D_t \left(\boldsymbol{u}^{n,K}\right) \\
\boldsymbol{u}^{n+1} &= \boldsymbol{u}^{n,K+1} + (\Delta t - (K+1)\delta t) \cdot \frac{\boldsymbol{u}^{n,K+1} - \boldsymbol{u}^{n,K}}{\delta t}
\end{aligned}
$$

Note

$$
\frac{\boldsymbol{u}^{n,j+1} - \boldsymbol{u}^{n,j}}{\delta t} = D_t \left(\boldsymbol{u}^{n,j}\right) = \boldsymbol{k}_j
$$

Write this as Runge-Kutta scheme with evaluations $\boldsymbol{k}_j$

Projective Integration as Runge-Kutta method

$$
\begin{aligned}
\boldsymbol{k}_0 &= D_t\left(\boldsymbol{u}^{n,0}\right) \\
\boldsymbol{k}_1 &= D_t\left(\boldsymbol{u}^{n,1}\right) = D_t\left(\boldsymbol{u}^{n,0} + \delta t \boldsymbol{k}_0\right) \\
\boldsymbol{k}_2 &= D_t\left(\boldsymbol{u}^{n,2}\right) = D_t\left(\boldsymbol{u}^{n,0} + \delta t \boldsymbol{k}_0 + \delta t \boldsymbol{k}_1\right) \\
&\vdots \\
\boldsymbol{k}_K &= D_t\left(\boldsymbol{u}^{n,K}\right) = D_t\left(\boldsymbol{u}^{n,0} + \delta t \sum_{j=0}^{K-1} \boldsymbol{k}_j\right) \\
\boldsymbol{u}^{n+1} &= \boldsymbol{u}^{n,0} + \Delta t\left(\sum_{j=0}^{K-1} \frac{\delta t}{\Delta t}\boldsymbol{k}_j + \frac{\Delta t - K\delta t}{\Delta t}\boldsymbol{k}_K\right)
\end{aligned}
$$

Projective Integration as Runge-Kutta method

$$\begin{aligned}
\boldsymbol{k}_l &= D_t\left(\boldsymbol{u}^{n,l}\right) = D_t\left(\boldsymbol{u}^{n,0} + \Delta t \sum_{j=0}^{l-1} \frac{\delta t}{\Delta t}\boldsymbol{k}_j\right) \\
\boldsymbol{u}^{n+1} &= \boldsymbol{u}^{n,0} + \Delta t\left(\sum_{j=0}^{K-1} \frac{\delta t}{\Delta t}\boldsymbol{k}_j + \frac{\Delta t - K\delta t}{\Delta t}\boldsymbol{k}_K\right)
\end{aligned}$$

Write Projective Integration in Butcher tableau

$$\begin{array}{c|c} \mathbf{c} & A \\ \hline & \mathbf{b}^{\mathbf{T}} \end{array} = \begin{array}{c|ccccc}
0 & 0 & & & \\
1 \cdot \frac{\delta t}{\Delta t} & \frac{\delta t}{\Delta t} & 0 & & \\
\vdots & \vdots & \ddots & \ddots & \\
K \cdot \frac{\delta t}{\Delta t} & \frac{\delta t}{\Delta t} & \dots & \frac{\delta t}{\Delta t} & 0 \\
\hline
 & \frac{\delta t}{\Delta t} & \dots & \frac{\delta t}{\Delta t} & 1 - K\frac{\delta t}{\Delta t}
\end{array}$$

Projective integration does $K+1$ small FE steps and one extrapolation

Projective RK scheme (PRK) [Lafitte et al., 2017]

Use standard RK scheme $\begin{array}{c|c} \mathbf{c} & A \\ \hline & \mathbf{b}^T \end{array}$ $\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t \sum_{j=1}^{s} b_j \boldsymbol{k}_j$

$$\boldsymbol{k}_j = D_t\left(\boldsymbol{u}^n + \Delta t \sum_{l=1}^{s} a_{jl} \boldsymbol{k}_l\right), \quad j = 1, \dots, s$$

Replace each time derivative $\boldsymbol{k}_l$ by an inner integrator and a time derivative estimate

$$\boldsymbol{u}_l^{n,k+1} = \boldsymbol{u}_l^{n,K} + \delta t D_t\left(\boldsymbol{u}_l^{n,K}\right), \quad 0 \leq k \leq K$$

$$k_l = \frac{\boldsymbol{u}_l^{n,K+1} - \boldsymbol{u}_l^{n,K}}{\delta t} = D_t\left(\boldsymbol{u}_l^{n,K}\right)$$

Stability of Projective Integration

Dahlquist test equation

$$\partial_t u = \lambda u, \quad \lambda < 0$$

$$u^{k+1} = \tau\left(\lambda \delta t\right) u^k$$

Amplification factor $\tau\left(\lambda \delta t\right)$.
Stability requires

$$\left|\tau\left(\lambda \delta t\right)\right| \leq 1$$

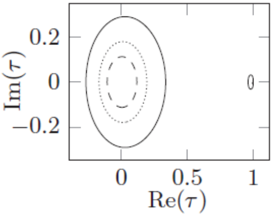
Forward Euler: $u^{n+1} = u^n + \Delta t \lambda u^n = (1 + \Delta t \lambda) u^n$

$$\Rightarrow \tau^{FE} = (1 + \Delta t \lambda)$$

Stability of Projective Integration

Stability region (PFE): $\mathcal{D}^{PFE} = D\left(1 - \frac{\delta t}{\Delta t}, \frac{\delta t}{\Delta t}\right) \cup D\left(0, \left(\frac{\delta t}{\Delta t}\right)^{1/K}\right)$

- First part corresponds to quickly damped modes
- Second part corresponds to slowly decaying modes



Choosing δt properly leads to accurate solution of slow modes while maintaining stability of fast modes.

Stability of Projective Integration, [Lafitte et al., 2017]

If inner integrator is stable PRK parameters only need to fulfill standard RK stability conditions.

$$\begin{array}{c|c} \mathbf{c} & A \\ \hline & \mathbf{b}^T \end{array}$$

Stability regions of lower-order methods are contained within those of higher-order methods.

Parameters for PFE will be valid for PRK.

Stability of Projective Integration, [Kevrekidis, 2014]

Projective Integration does not have unlimited stability w.r.t. λ.

For larger λ, more inner time steps K are necessary

$$\text{for } k = 0, 1, \dots, K : \quad \boldsymbol{u}^{n,k+1} = \boldsymbol{u}^{n,k} + \delta t \boldsymbol{D}\left(\boldsymbol{u}^{n,k}\right)$$

Equivalently, smaller extrapolation steps Δt could be chosen

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^{n,K+1} + (\Delta t - (K+1)\delta t) \cdot \frac{\boldsymbol{u}^{n,K+1} - \boldsymbol{u}^{n,K}}{\delta t}$$

Still speedup w.r.t. standard Forward Euler method.

Stability of Projective Integration, [Kevrekidis, 2014]

Projective Integration does not have unlimited stability w.r.t. λ.

For larger λ, more inner time steps K are necessary

$$\text{for } k = 0, 1, \ldots, K : \quad \boldsymbol{u}^{n,k+1} = \boldsymbol{u}^{n,k} + \delta t \boldsymbol{D}\left(\boldsymbol{u}^{n,k}\right)$$

Equivalently, smaller extrapolation steps Δt could be chosen

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^{n,K+1} + (\Delta t - (K+1)\delta t) \cdot \frac{\boldsymbol{u}^{n,K+1} - \boldsymbol{u}^{n,K}}{\delta t}$$

Still speedup w.r.t. standard Forward Euler method.

For kinetic equations, we can still get unlimited stability.

Parameter choice, [Lafitte et al., 2017]

$$\partial_t f^\epsilon + \frac{v}{\epsilon^\gamma}\partial_x f^\epsilon = \frac{Q\left(f^\epsilon\right)}{\epsilon^{\gamma+1}}$$

$\gamma = 0$ hydrodynamic scaling,
$\gamma = 1$ diffusive scaling

Parameter choice

- $\Delta t = \mathcal{O}\left(\Delta x\right)$ for hydrodynamic limit
- $\delta t = \mathcal{O}\left(\epsilon\right)$ for hydrodynamic limit
- $\Delta t = \mathcal{O}\left(\Delta x^2\right)$ for diffusive limit
- $\delta t = \mathcal{O}\left(\epsilon^2\right)$ for diffusive limit
- K is small number, typically $K \leq 3$

Parameter Example, [Lafitte et al., 2017]

Hyperbolic scaling, DVM, linearized BGK, third-order upwind discretization

$$\delta t = \epsilon,$$

$$K \geq 2,$$

$$\Delta t \leq \left(\frac{3\Delta x}{4c_0}, \frac{3\Delta x}{8} \right)$$

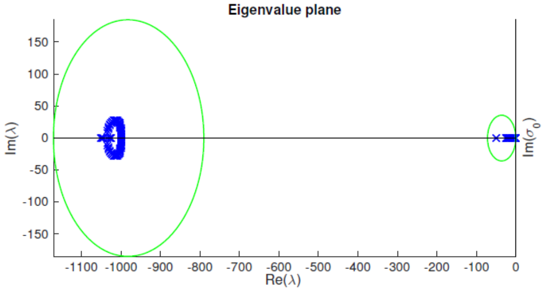
where c_0 is related to the DVM discretization.

Δt might be larger, this is only an estimate.

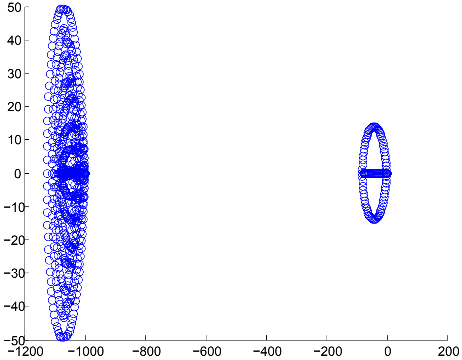
Projective Integration

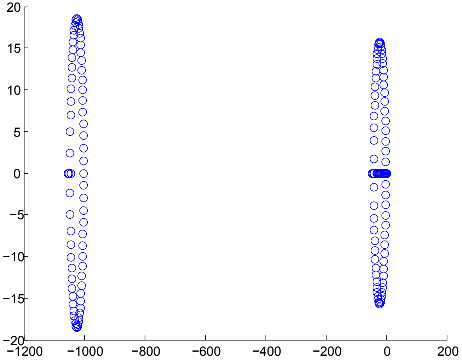
Properties of PI

- Small number of inner steps sufficient, K independent of ϵ
- Inner and outer integrators can be RK schemes $\Rightarrow$ high-order
- Works best for spectral gap



Spectral gap in shock tube

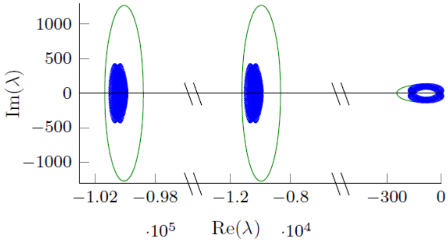




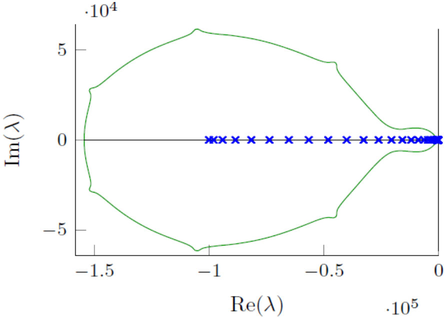
Moment models are ideally suited for Projective Integration

No classical spectral gap

Discrete values for ϵ



Continuous $\epsilon \sim \rho(t,x)$

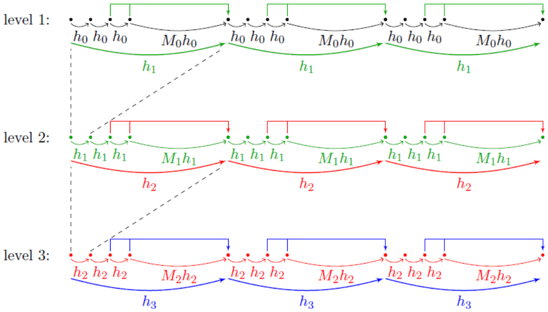


- adjust number of inner time steps: $K = \mathcal{O}(\log(1/\epsilon))$
- prevent stability region split $\Rightarrow$ [0,1]-stable method $\Rightarrow$ Δt limited by ϵ
- **Telescopic Projective Integration (TPI)**

Telescopic Projective Integration [Melis, Samaey, 2018]

Idea:

Nested Projective Integration



Integrators at different levels

Innermost integrator

- needs to capture fastest components and damp them
- higher-order methods result in stability restriction for other levels
- simply use Forward Euler (FE)

Projective integrators

- outermost dominates accuracy of the scheme
- simplest version Forward Euler (TPFE)
- higher-order (outermost) Runge Kutta (TPRK)

Parameter selection [Melis, Samaey, 2018]

- number of steps at level l: K_l
- time step size at level l: δt_l
- extrapolation size at level l: M_l

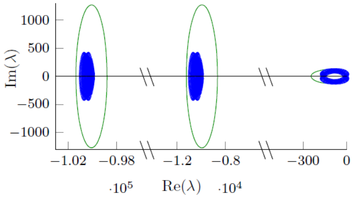
Choice based on spectrum of the respective method

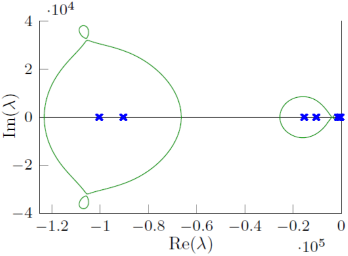
Parameter selection

$$\partial_t \boldsymbol{u} + \boldsymbol{A}_M \partial_x \boldsymbol{u} = -\frac{\omega(x)}{\epsilon} \boldsymbol{Q} \boldsymbol{u}$$

discrete ω levels

- Capture each eigenvalue cluster with one level
- Merge adjacent clusters to one
- Possibility to construct a connected stability region, depends on ϵ



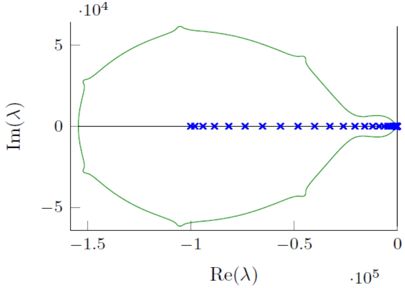


Parameter selection

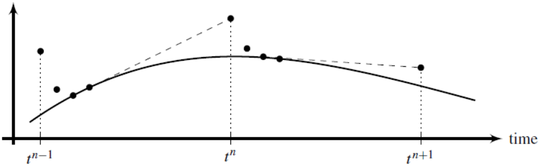
$$\partial_t \boldsymbol{u} + \boldsymbol{A}_M \partial_x \boldsymbol{u} = -\frac{\omega(x)}{\epsilon} \boldsymbol{Q} \boldsymbol{u}$$

continuous ω

- Need to construct a connected stability region, depends on ϵ
- $[0, 1]$-stable method
- Number of levels depends on $\log(1/\epsilon)$



Summary of Projective Integration



PI

- PRK, TPI extensions
- AP for const relaxation
- Almost AP for continuous relaxation
- TPI speedups w.r.t. FE

Further work on Projective Integration

Past work [Melis, Samaey, 2018], [Lafitte et al., 2017]

- only for DVM models
- only for BGK and linearized Maxwellians

Next steps

- implement PI for moment models
- extension TPRK and TPI straightforward
- use PI for other models/collision operators

Further work on Projective Integration

Past work [Melis, Samaey, 2018], [Lafitte et al., 2017]

- only for DVM models
- only for BGK and linearized Maxwellians

Next steps

- implement PI for moment models
- extension TPRK and TPI straightforward
- use PI for other models/collision operators

Thank you for your attention!

References

Efficient asymptotic-preserving (AP) schemes for some multiscale kinetic equations,

S. Jin, *J. Sci. Comput.* **21(2)**, 441-454, 1999

Projective methods for stiff differential equations: problems with gaps in their eigenvalue spectrum,

C.W. Gear, I.G. Kevrekidis, , *J. Sci. Comput.* **24(4)**, 1091-1106, 2003

Implicit-Explicit Runge-Kutta schemes and applications to hyperbolic systems with relaxations

L. Pareschi, G. Russo, *J. Sci. Comput.* **25**, 129-155, 2005

A high-order relaxation method with projective integration for solving nonlinear systems of hyperbolic conservation laws,

P. Lafitte, W. Melis, G. Samaey, *J. Comput. Phys.* **340**, 1-25, 2017

Telescopic Projective Integration for Linear Kinetic Equations with Multiple Relaxation Times,

W. Melis, G. Samaey, *J. Sci. Comput.* **76**, 697-726, 2018